

Index Optimization with Combination of Text Categorization using String Vector based KNN

Taeho Jo
President
Alpha AI Research
Cheongju, South Korea
tjo018@naver.com

Abstract—In this research, we propose the string vector based KNN version for automating the index optimization with its combination of the text classification. In the previous work, the string vector based KNN version was proposed as an approach to the index optimization, but the domain should be presented as a tag, together with the text, manually. In this research, the string vector-based version which receives a string vector instead of a numerical vector is adopted as the approach to the text classification and the keyword extraction. In the proposed system, only a text without its domain is given as the input, its domain is automatically given through the text classification, and each word in the text is classified into one of the three categories by the classifier which corresponds to the domain. The goal of this research is to reinforce the automation of the index optimization by introducing the text classification, and to improve the performance of both tasks, using the string vector based KNN algorithm.

I. INTRODUCTION

The index optimization is the process of optimizing the text index for maximizing the information retrieval performance by the index expansion and the index filtering. The words in the given text are defined as the three classes, expansion, inclusion, and removal, and the index optimization is converted into the classification task where each word is classified into one of the three categories. The process of index optimization is to index a text into words, classify each word into one of the three categories, and treat the classified words depending on their categories. To words which are classified with expansion, associative words from external sources are added as the index expansion, the words which are classified with inclusion are taken by themselves, and the words which are classified into removal are excluded. Because the classification which is mapped from the index optimization is a domain dependent task where a same entity may be classified differently depending on the domain, it should be presented as a tag for executing the index optimization.

This research is motivated by the fact of knowing the domain of the input text in advance for optimizing indexes from it. The index optimization is set as an independent classification task domain by domain. We need to know the domain of the input text for selecting a corresponding

classifier. It is possible to present a text without its domain by automatically assigning the domain to the input text through the text classification. The solution to the problem is the connection of the index optimization which is covered in this research with the text classification.

In this research, we adopt the string vector based KNN algorithm for implementing the index optimization which is combined with the text classification. We define the semantic similarity between two string vectors as a semantic operation and modify the KNN algorithm into the string vector-based version. It is applied to the text classification which assigns a domain to an input text. The input text is indexed into words, and the modified KNN algorithm is applied to the classification of words into one of the three categories. This research is intended to automate the process of labeling a text which is given as the input for the index optimization with its own domain.

Let us mention some benefits from this research. The problems in encoding words and texts into numerical vectors are avoided by encoding them into string vectors. The performance of the index optimization and the text classification is improved by adopting the string vector based KNN algorithm. We do not need to decide the domain manually to the input text for using the index optimization system. In this research, we upgrade the system by automating the process of assigning the domain to the input text.

Let us mention some remaining tasks as further research. We may define and characterize mathematically more semantic operations on string vectors. Texts and words may be encoded into compound structured forms. Other machine learning algorithms and deep learning algorithms are modified into string vector-based versions. The index optimization system may be implemented as a real version or a library module by adopting the string vector based KNN algorithm.

II. PROPOSED APPROACH

A. Similarity Metric

This section is concerned with the string vector as the representation of a word and the similarity between string vectors. The string vector is defined as a finite ordered set

of strings; in the vector, the numerical values are replaced by the strings. It is required to define a similarity matrix for computing the similarity between elements. The similarity between string vectors is the average over one-to-one similarities between their elements. This section is intended to describe the process of encoding a word into a string vector and the process of computing the similarity between string vectors.

The process of encoding words into string vectors is illustrated in Figure 1. In the word-text association, a text collection is constructed by gathering texts, and each word is associated with its own texts based on their information. The string vector features are defined as symbolic forms manually in the feature definition. In the feature value assignment, the string vector which represents a word is filled with text identifiers which correspond to the features. Refer to [1] for studying the encoding process in more detail.

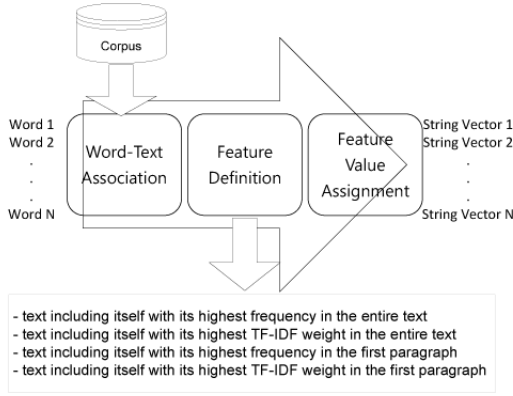


Figure 1. Process of Encoding Words into String Vectors

The similarity matrix which is the basis for computing the similarity between string vectors is illustrated in figure 00. In the matrix, each row and each column correspond to text identifiers, and each element is the similarity between texts, d_i and d_j , as shown in equation (1),

$$s_{ij} = \text{sim}(d_i, d_j) \quad (1)$$

The similarity between two texts, d_i and d_j , is computed by equation (2),

$$\text{sim}(d_i, d_j) = \frac{2|D_i \cap D_j|}{|D_i| + |D_j|} \quad (2)$$

where D_i is the set of words in the text, d_i , and D_j is the set of words in the text, d_j . The value of similarity between texts, d_i and d_j , is always given as a normalized value between zero and one, as shown in equation (3)

$$0 \leq \text{sim}(d_i, d_j) \leq 1 \quad (3)$$

The similarity matrix is used for computing the similarity between string vectors which represent words as a reference table.

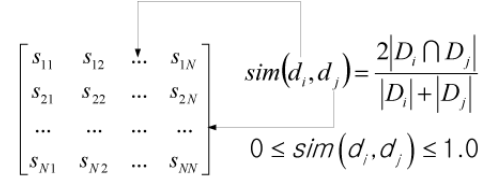


Figure 2. Semantic Similarity Matrix

Let us mention the process of computing the similarity between string vectors as a semantic similarity between words. The two string vectors which represent words are notated by $\text{str}_1 = [d_{11} \ d_{12} \ \dots \ d_{1d}]$ and $\text{str}_2 = [d_{21} \ d_{22} \ \dots \ d_{2d}]$. The similarity between two string vectors is computed by equation (4),

$$\text{sim}(\text{str}_1, \text{str}_2) = \frac{1}{d} \sum_{i=1}^d \text{sim}(d_{1i}, d_{2i}). \quad (4)$$

The similarity between elements, $\text{sim}(d_{1i}, d_{2i})$, is taken from the similarity which was mentioned above. The value which is generated by equation (4) is always given as a normalized value between zero and one as shown in equation (5),

$$0 \leq \text{sim}(\text{str}_1, \text{str}_2) \leq 1. \quad (5)$$

Let us make some remarks on the process of encoding words into string vectors and computing the similarity between words. A word is encoded into a string vector by the word-text association, the feature definition, and the feature value assignment. The similarity matrix is defined as the basis for computing the similarity between string vectors. The similarity is computed by equation (4). In the future research, we consider representing a word into multiple string vectors.

B. String Vector based KNN Algorithm

This section is concerned with the string vector based KNN algorithm. It was initially proposed as the approach to the text summarization by Jo in 2018 ?? . The similarity metric between string vectors which was described in the previous section is used for computing the similarity between a novice string vector and a training example. The labels of its nearest neighbors are voted with their identical weights for deciding the label of a novice input. This section is intended to describe the initial version of KNN algorithm from which its variants are derived.

The process of computing the similarity between a novice item and a training example is illustrated in Figure 00. The labeled words which are gathered as samples are encoded into string vectors, and they are notated by a set, $Tr = \{\text{str}_1, \text{str}_2, \dots, \text{str}_N\}$. A novice word is encoded into a string vector notated by str . Its similarities with the string vectors which represent the sample words are computed by equation (4), as

$sim(\mathbf{str}, \mathbf{str}_1), sim(\mathbf{str}, \mathbf{str}_2), \dots, sim(\mathbf{str}, \mathbf{str}_N)$. Some training examples which are most similar as the novice input are selected as its nearest neighbors.

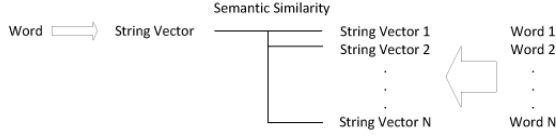


Figure 3. Similarities of Novice Input with Training Examples

In Figure 4, the process of selecting nearest neighbors by the ranking selection is illustrated. The training examples are sorted by the descending order of their similarities, after computing their similarities with a novice example. The training examples with their higher similarities with a novice example are selected as its nearest neighbors, as expressed in equation (6),

$$Nr = Select_k(Tr, \mathbf{str}), Nr \subseteq Tr \quad (6)$$

The set of nearest neighbors, Nr , is composed with elements as expressed in equation (7),

$$Nr = \{\mathbf{str}_1^{near}, \mathbf{str}_2^{near}, \dots, \mathbf{str}_k^{near}\} \quad (7)$$

The elements in the set, Nr , are used for voting their labels in the next step.

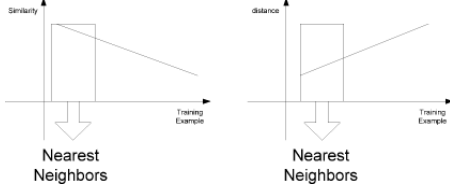


Figure 4. Selection of Most Similar or Least Distant Training Examples as Nearest Neighbors

The process of voting the labels of nearest neighbors for decoding the label of a novice word is illustrated in Figure 5. The predefined categories are notated by $c_1, c_2, \dots, c_m$, and the label of a nearest neighbor is notated by $c_j = label(\mathbf{str}_i^{near})$. The number of nearest neighbors which belong to the category, c_j , is notated by $Count(Nr, c_j)$. The label of the novice item, $\mathbf{str}$, is decided by equation (8),

$$label(\mathbf{str}) = \underset{j=1}{\operatorname{argmax}}^m Count(Nr, c_j) \quad (8)$$

The process of deciding the label of a novice item by equation (8), is called voting.

Let us make some remarks on the string vector based KNN algorithm. It computes the similarities of a novice item with the training examples. The training examples with their highest similarities with the novice item are selected as its nearest neighbors. The label of the novice item is decided by voting the labels of its nearest neighbors. The ranking



Figure 5. Voting of Labels of Nearest Neighbors for Deciding Label of Novice Input

selection is adopted for selecting the nearest neighbors from training examples, in this version.

C. System Architecture

This section is concerned with the index optimization system which adopts the string vector based KNN algorithm. In Section II-B, we described the proposed version of KNN algorithm as the approach to the index optimization. It is viewed as a classification task which classifies a word into one of the three categories. This section is intended to describe the index optimization system with respect to its function and its architecture.

The process of collecting the same words and encoding them into string vectors for implementing the index optimization system is illustrated in Figure 6. The index optimization is interpreted into a domain dependent classification; even a same word is classified differently, depending on the domain. For each domain, an independent classification task is defined. The several domains are decided, and the words which are labeled with one of the three categories exclusively in each domain. It is required to present the domain from a text which is given as the input for doing this task.

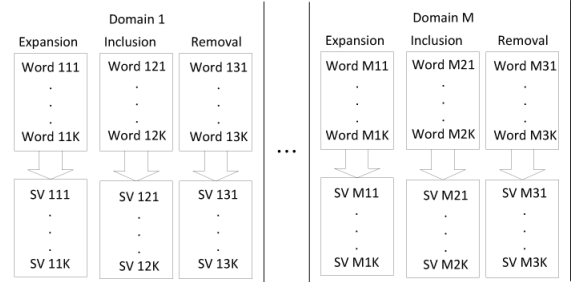


Figure 6. Preparation of Training Examples

The entire architecture of the proposed index optimization system is illustrated in Figure 7. A text is given as the input, and words are extracted from it in the indexing module. In the encoding module, the sample words in the groups, expansion, inclusion, and removal and ones which are indexed from the text are encoded into string vectors. In the similarity computation module and the voting module, the words which are indexed from the text are classified into one of the three categories. The words which are classified into removal are discarded in this system.

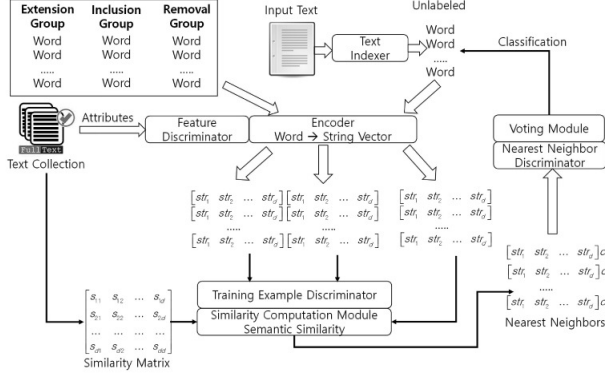


Figure 7. System Architecture

The execution process of the proposed system is illustrated as a block diagram in Figure 8. Sample words which are labeled with one of the three categories are collected within a domain, and they are encoded into string vectors. An input text is indexed into a list of words, and they are also encoded into string vectors. The nearest neighbors are selected by the similarity computation, and the label is decided by voting ones of their nearest neighbors for each word. Ones which are classified into expansion require their associated words from external sources, ones which are classified into inclusion are preserved, and ones which are classified into removal are excluded.

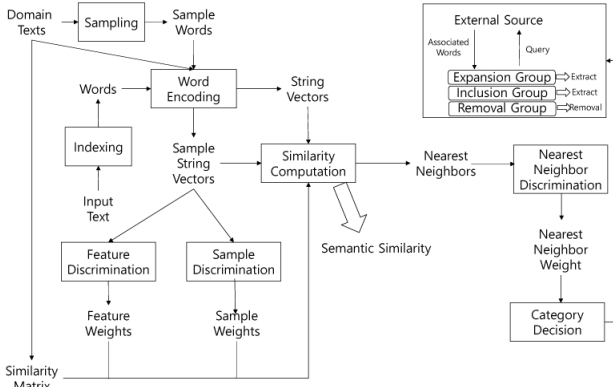


Figure 8. System Flow

Let us make some remarks on the proposed system which is illustrated in Figure 7 as its architecture. The index optimization is defined as a multiple classification task where each word is classified into expansion, inclusion, or removal. The words are encoded into string vectors instead of numerical vectors, and they are classified directly. The words which are classified into removal are excluded from index of the input text. In implementing the system as its complete version, we need to add the module which retrieve more words which are relevant to ones which are labeled

with expansion.

D. Connection with Text Classification

This section is concerned with the connection of the index optimization with the text classification. In the previous section, we mentioned the index optimization system as an instance of domain dependent classification. The domain is automatically decided for nominating a classifier by connecting the keyword with the text classification. The KNN algorithm which was described in Section II-B is used for implementing the text classification. This section is intended to describe the connection of the index optimization system with the text classification for automating the decision of the domain.

The system architecture of the text categorization system which consists of the encoding module, the similarity computation module, and the voting module is illustrated in Figure 9. The encoding module is for encoding texts into string vectors by the process which is described in Section II-A. The similarity computation module is for computing the similarities between string vectors which represent a novice text and a sample text and selecting ones with their highest similarities as the nearest neighbors. The voting module is for deciding the label of the novice item by voting the labels of the nearest neighbors. This system used for automating deciding the domain of the input text.

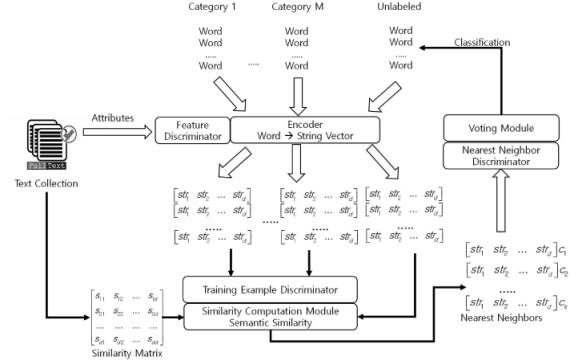


Figure 9. Text Categorization System

The connection of the index optimization with the text classification is illustrated in Figure 10. The texts each of which is labeled with its own domain are prepared as the sample texts, and the approach to the text classification is trained with them. A domain is assigned to a novice text by classifying it into a domain automatically. The novice text is provided with its domain to the index optimization system which is described in Section II-C. The role of the text classification system is to nominate the classifier which corresponds to the domain for the index optimization, automatically.

The process of executing the index optimization, connecting it with the text classification is illustrated in Figure 11.

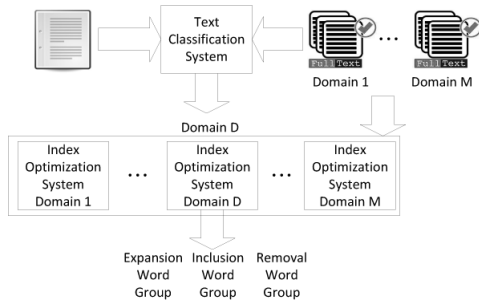


Figure 10. Index Optimization System connected with Text categorization

The sample texts each of which is labeled with its own domain are prepared, and the sample words each of which is labeled with one of the three categories are prepared in each domain. A novice text is classified into one among the predefined domains, and the classifier which corresponds to the domain. The novice text is indexed into a list of words, and each word is classified into one of the three categories. The labeled words are the final output from this system; the domain which is assigned to the input text is the guide for selecting a classifier.

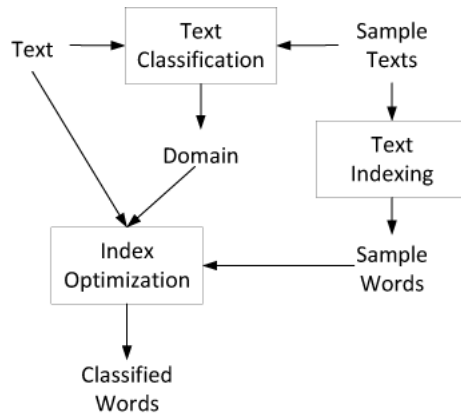


Figure 11. System Flow of Index Optimization connected with Text Categorization

Let us make some remarks on the connection of the index optimization with the text classification. The text classification is applied to the automatic decision of the input text domain. The role of text classification is to decide the domain for nominating a classifier, and the role of the index optimization is to classify words into expansion, inclusion, or removal. In the execution flow, the input text is classified into a domain, it is indexed into a list of words, and each word is classified into one of the three categories. We consider connecting the text classification as the main task the index optimization as the opposite to what is proposed in this research.

REFERENCES

- [1] T. Jo, "Automatic Text Summarization using String Vector based K Nearest Neighbor", 6005-6016, Journal of Intelligent and Fuzzy Systems, 35, 2018.
- [2] T. Jo, "String Vector based Version of K Nearest Neighbor for Index Optimization in Current Affairs", 47-50, The Proceedings of International Conference on Applied Cognitive Computing, 2018.